6 Filas

" Vamos mudar a nossa atitude tradicional de construir programas. Em vez de imaginar que a nossa tarefa principal é instruir um computador a fazer o que queremos, deveremos começar a pensar que a nossa tarefa é explicar a seres humanos o que nós queremos que o computador faça "

- Donald E. Knuth -

Sumário:

6.1- Conceitos

Uma **fila**(**Queue**) é uma lista linear onde as inserções são feitas numa extremidada, denominada por fundo, e as remoções são feitas numa outra extremidade, denominada por frente. Como o primeiro elemento a entrar na fila é o primeiro a sair, estas estruturas são denominadas como listas **FIFO**. Este termo deve-se a seguinte frase:*"first-in , first-out".*

Esta propriedade torna a fila uma ferramenta ideal para o desenvolvimento de diversas aplicações na informática. Por exemplo, a estrutura apropriada para gerir os processos de impressão num sistema multiusuários, onde sómente um pedido pode ser atendido de cada vez, é uma fila, denominada por SPOOL.

6.2- Tipo Abstracto de Dados

Um possível conjunto de operações para criar um tipo abstracto de dados fila é descrito pelas operações:

- Criar uma fila vazia;
- Verificar se a fila está vazia;
- Inserir o elemento do fundo da fila "enqueue" ;
- Remover o elemento da frente da fila " dequeue";
- Consultar o conteudo de um elemento
- Contar o número de elementos;

6.3 - Fila Estática

Uma forma muito simples de implementar esta estrutura, consiste em fixar o ínicio da fila na primeira posição do vector. Com essa estratégia, as inserções fazem com que o contador fundo seja incrementado em uma unidade, mas, as remoções, requerem a deslocação de todos os elementos para a primeira posição. Essa operação não é eficiente. Para optimiza-la, não devemos fixar o ínicio da fila na primeira posição. Isso quer dizer, que esta estrutura deverá possuir dois índices: **frente** para as remoções e **fundo** para as inserções.

6.3.1- Códigos de Erros

```
#define  OK              0     // Operação realizada com sucesso
#define  QUEUE_FULL      1     //  Fila  cheia
#define  QUEUE-EMPTY     2     //  Fila vazia
#define NO_MEMORY        3     //  Memória insuficiente
```

6.3.2- Estrutura de Dados

```
#define TAMANHO  100

typedef struct
{
     int  valor;
}TItem;

typedef struct
{
    Titem item[TAM];
     int frente;
     int  fundo;
} TFila;

typedey  enum  {FALSE= 0, TRUE= 1} Boolean;
```

6.3.3- Implementação das Operações

De forma analoga a lista linear e pilha, inicializar uma fila, consiste em associar aos seus índices frente e fundo um valor anterior ao ínicio do primeiro posição do vector.

```
/*-------------------------------------------------------------------------------------------------
Objectivo: Criar uma estrutura de dados do tipo fila
Parâmetro Entrada: Um vector
Parámetro de Saída: Fila
-------------------------------------------------------------------------------------------------*/
void  inicializarFila (TFila *fila)
{
    fila->frente = -1;
    file->fundo = -1;
}
```

A operação para determinar se uma fila está vazia, consiste em verificar se ela não possui elementos. Deixamos este operador como exercício. De forma analoga, omitiremos a operação para verificar se uma fila está cheia.

A operação para inserir um elemento numa fila, consiste em verificar em primeiro lugar, se ela não está cheia. Se essa condição for verdadeira, o processo de inserção consiste em deslocar o índice do fundo para uma posição à direita (adicionar uma únidade) e em seguida inserir o elemento nessa posição.

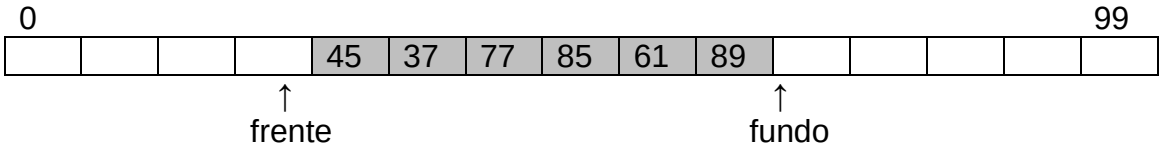
```
/*-------------------------------------------------------------------------------------------------
Objectivo: Inserir um elemento no fundo da fila
Parâmetro Entrada: Um elemento qualquer e uma fila
Parâmetro de Saída: Fila actualizada
Valor de Retorno: Código de erro ( QUEUE_FULL ou OK)
-------------------------------------------------------------------------------------------------*/
```

```
int  inserirFila (int x, TFila *fila)
{
   if ( cheiaFila(fila) )
      return QUEUE_FULL;
   else {
            fila->fundo++;
            fila->item[fila->fundo] =  x;
            return OK;
         }
}
```

A operação para remover um elemento numa fila, consiste em verificar em primeiro lugar se ela não está vazia. Se essa condição for verdadaira, o processo de remoção consiste em deslocar o índice da frente para uma posição á direita (adicionar uma únidade) e em seguida, guardar numa variável de memória o elemento que se pretende remover. Devido as movimentação dos índices frente e fundo, poderemos chegar à uma situação em que frente é igual a fundo, nesse caso deve-se inicializar a fila.

```
/*-------------------------------------------------------------------------------------------------------
Objectivo: Remover um elemento na frente da fila
Parâmetro Entrada: Fila
Parâmetro de saída: Fila actualizada, o conteúdo do elemento removido
Valor de Retorno: Código erro ( QUEUE_EMPTY ou OK)
-------------------------------------------------------------------------------------------------------*/
int removerFila (TFila *fila, int *x)
{
  if ( vaziaFila(fila) )
     return QUEUE_EMPTY;
  else {
            fila->frente++;
            *x= fila->item[fila->frente];
             if  (fila->frente ==  fila-> fundo)
                 inicializar(&fila);
            return OK;
         }
}
```

Não é demais salientar que o índice frente contêm a posição do elemento que antecede o primeiro elemento da fila, enquanto fundo contêm a posição do último elemento inserido na fila. Em termos gráficos:

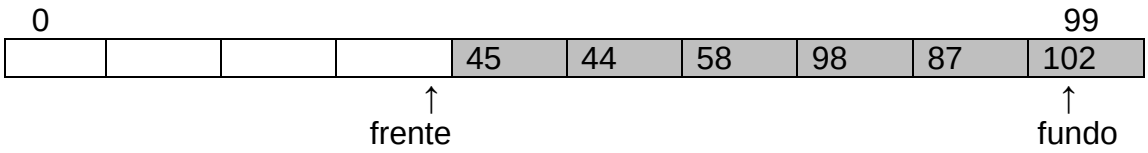


A operação para determinar o tamanho de uma fila, consiste em devolver o seu número de elementos. Para isso, basta efectuar seguinte a operação álgebrica fundo-frente.

```
/*-----------------------------------------------------------------------------------------------
Objectivo: Determinar o número de elementos da fila
Parâmetro Entrada: Fila
Valor de Retorno: Número inteiro maior ou igual a zero
-----------------------------------------------------------------------------------------------*/
int  TamanhoFila (Tfila  fila);
{
   return   (fila.fundo - fila.frente);
}
```

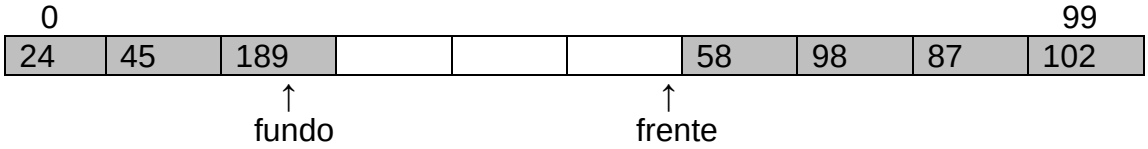
De forma similar a pilha, o acesso a qualquer elemento no meio de uma fila só é possível se forem removidos todos os elementos que estão a sua frente, ou seja, os elementos que estão entre o índice frente o índice que faz referencia ao antecessor do elemento que se quer aceder.

Vamos em seguida analisar com algum detalhe, a eficiência dos algoritmos de inserção e remoção vistos anteriormente. Aparentemente funcionam. O que acontece se fundo = TAMANHO - 1 e desejamos inserir mais um elemento?



O procedimento de inserção que estudamos tem o seguinte comportamento. Se por um lado, a condição de fila cheia for determinada pela comparação do índice fundo com TAMANHO-1, o elemento não será inserido, apesar de termos espaço livre. Por outro, se a condição de fila cheia for determinada pela conjunção de condições, frente igual à -1 e fundo igual à TAMANHO - 1, o procedimento vai inserir o elemento num endereço de memória que não pertence ao vector. Como a linguagem C não controla os limites do vector o nosso programa dar-nos-á resultados imprevisíveis

Uma solução eficiênte consiste em permitir que o índice fundo possa voltar para a primeira posição do vector. Essa implementação é conhecida como fila circular. Mostramos, em seguida, um exemplo ilustractivo onde fundo é menor do que frente.



6.4 - Fila Circular

Um método eficiente de gerir todo o espaço disponível no vector, consiste na implementação das operações sobre o tipo abstracto de dados fila, com a aritmética modular. Essas operações transformam esse tipo de dados no tipo abstracto de dados fila circular, que declaramos em seguida:

6.4.1- Estrutura das Dados

```
#define TAMANHO  100

typedef struct
{
      int  valor;
}TItem;

typedef struct
}
     Titem item[TAM];
      int frente, fundo, Tamanho;
} TFilaCirc;

typedey  enum  {FALSE= 0, TRUE=1} Boolean;
```

6.4.2- Implementação das Operações

A operação para inicializar uma fila circular consiste em associar os seus índices ao meio da fila e colocar zeros numa variável que determina o número de elementos inseridos.

```
/*-------------------------------------------------------------------------------------------------------
Objectivo: Criar uma estrutura de dados do tipo fila circular
Parâmetro de Entrada: vector fila
Parâmetro saída: Fila circular
-------------------------------------------------------------------------------------------------------*/
void  inicializarFilaCircular(TFilaCirc *fila)
{
   fila->frente = TAMANHO/2;
   fila->fundo = TAMANHO/2;
   fila->tamanho = 0;
}
```

A operação para determinar se uma fila circular está vazia fica como exercício.

A operação para determinar se uma fila circular está cheia, consiste em verificar se todos os elementos do vector estão preenchidos. Para isso, basta verificar se a variável que guarda o número de elementos é igual a dimensão do vector que armazena essa fila.

```
/*-------------------------------------------------------------------------------------------------------
Objectivo: Verificar se a fila circular está cheia
Parâmetro Entrada: Fila circular
Valor de Retorno: Verdadeiro ou falso
-------------------------------------------------------------------------------------------------------*/
Boolean cheiaFilaCircular (TFilaCirc fila)
```

```
{
   return  (fila.tamanho == TAMANHO);
}
```

A operação para inserir um elemento numa fila circular, consiste em verificar em primeiro lugar, se ela não está cheia. Se essa condição for verdadeira, o processo de inserção consiste em deslocar o índice do fundo para à direita com base na aritmética modular. Em seguida, inserir o elemento nessa posição. Quando o índice do fundo fizer referência ao último elemento do vector, o mecanismo de aritmética modular insere esse elemento na primeira posição da fila.

```
/*-----------------------------------------------------------------------------------------------------
Objectivo: Inserir  um elemento no fundo de uma fila circular
Parâmetro Entrada: Um elemento qualquer e uma fila circular
Parametro Saída : Fila circular actualizada
Valor Retorno: código de erro (QUEU_FULL ou OK)
-----------------------------------------------------------------------------------------------------*/
int inserirFilaCircular (int x, TFilaCirc *fila)
{
   if ( cheiaFilaCircular(fila) )
      return QUEUE_FULL;
   else  {
            fila->fundo = (fila->fundo + 1) % TAMANHO;
            fila->tamanho++;
            fila->item[fila->fundo] =  x;
            return OK;
         }
}
```

A operação para remover um elemento numa fila circular, consiste em verificar em primeiro lugar, se a fila não está vazia. Se essa condição for verdadeira, o processo de remoção consiste em deslocar o índice da frente para á direita com base na aritmética modular. Em seguida, guardar numa variável de memória o elemento que se pretende remover. Quando o índice de frente fizer referência ao último elemento do vector, o mecanismo de aritmética modular remove o elemento que está na primeira posição desse vector.

```
/*-----------------------------------------------------------------------------------------------------
Objectivo: Remover o elemento da frente de uma fila circular
Parâmetro Entrada: Fila circular
Parâmetro de Saída:Fila circular actualizada, conteudo do elemento removido
Valor de Retorno: codigo de erro (QUEUE_EMPTY ou OK )
-----------------------------------------------------------------------------------------------------*/
int removerFilaCircular(TFilaCirc *fila, int  *x);
{
    if  ( vaziaFilaCircular(fila) )
        return QUEUE_EMPTY;
    else {
            fila->frente = (fila->frente + 1) % TAMANHO;
```

```
            fila->tamanho--;
            *x =  fila->item[fila->frente];
            return OK;
        }
}
```

6.5- Fila Dinâmica

O tipo abstracto de dados fila em organização dinâmica e fila em organização sequêncial possuem as mesmas operações.

6.5.1 - Códigos de Erro

```
#define NOT_FOUND      -1    // Item não existe
#define  OK             0    // Operação realizada com sucesso
#define  QUEUE_EMPTY    2    //  Fila vazia
#define NO_SPACE        5    // Não há espaço de memória
```

6.5.2 - Estrutura de Dados

```
typedef struct
 {
   int   chave;
   int   valor;
 }TInfo;

typedef struct apontador
 {
   TInfo   info;
   apontador  *prox;
 }TAtomo;
```

Declaramos uma fila em organização dinâmica com um ponteiro frente que faz referencia ao primeiro átomo da fila e um ponteiro fundo que faz referencia ao último átomo da fila.

```
typedef struct
  {
      TAtomo * frente;
      TAtomo  *fundo;
   } TFilaDinamic;

typedey  enum   { FALSE= 0, TRUE= 1 } Boolean;
```

6.5.3 - Implementação das Operações

De forma similar a lista ligada simples, inicializar uma fila, consiste em associar os ponteiros frente e fundo à um endereço nulo.

```
*/-------------------------------------------------------------------------------------------------------
Objectivo: Inicializar uma fila em organização dinâmica
Parâmetro Entrada  Uma fila
Parâmetro de saída:  fila dinâmica com ponteiros actualizados.
------------------------------------------------------------------------------------------------------- */
void   iniciaFilaDinamica (TFilaDinamic *fila)
{
   fila->frente = NULL;
   fila->fundo = NULL;
 }
```

Uma fila está vazia quando os ponteiros de frente e fundo fazem referência a um endereço nulo.

```
*/-------------------------------------------------------------------------------------------------------
Objectivo: Verificar se a fila está vazia
Parâmetro Entrada:  Uma fila
Valor de Retorno:  True ou False
------------------------------------------------------------------------------------------------------- */
 Boolean vaziaFilaDinamica (TFilaDinamic *fila)
{
   return (fila->frente == NULL  &&  fila->fundo == NULL);
}
```

A operação para inserir um átomo no fim da fila, consiste em solicitar um átomo à memória. Se essa operação não for realizada com sucesso, emitir o correspondente código de erro. No caso contrário, armazenar o conteudo que pretendemos inserir nesse átomo, colocar como sucessor desse átomo um endereço nulo e ligar esse átomo ao fim da fila. Mas se a fila estiver vazia, não é necessário ligar o novo átomo ao fim, basta associar o endereço do átomo extraído aos ponteiros de frente e fundo.

```
*/-------------------------------------------------------------------------------------------------------
Objectivo: Inserir um átomo com uma determinada informação no fim da fila
Parâmetro Entrada:  Uma fila e uma determinada informação
Parâmetro de Saída: Fila actualizada
Valor de Retorno: Código de erro ( NO_SPACE ou OK)
------------------------------------------------------------------------------------------------------- */
int  inserirFilaDinamica (TFilaDinamic *fila, TInfo x)
{
   TAtomo *pnovo = (TAtomo *) malloc( sizeof(TAtomo));
    if (pnovo == NULL)
       return NO_SPACE;
   else {
          pnovo->info = x;
          pnovo->prox = NULL;
          if ( vaziaFilaDinamica(fila) )
              {
                  fila->fundo = pnovo;
```

```
                fila->frente = pnovo;
            }
        else
            {
                fila->fundo->prox = pnovo;
                fila->fundo = pnovo;
             }
    }
}
```

A operação para remoção de um átomo no início da fila consiste em verificar se a fila está vazia. Se essa condição for verdadeira, devolver o correspondente código de erro. No caso contrário, criar um ponteiro auxiliar que faz referencia ao início da fila, guardar a informação que pretendemos remover e movimentar o ponteiro frente para o seu sucessor. Mas se a fila for unitária, inicializa-la. Para terminar, devolver o átomo removido à memória.

```
*/-------------------------------------------------------------------------------------------------
Objectivo: Remover o átomo que está no ínicio da fila
Parâmetro Entrada:  Uma fila
Parâmetro de Saída: Fila actualizada e a informação removida
Valor de Retorno: Código de erro (QUEUE_EMPTY ou OK)
------------------------------------------------------------------------------------------------- */
int removerFilaDinamica (TFilaDinamic *fila, TInfo *x)
{
  if ( vaziaFilaDinamica(fila))
     return QUEUE_EMPTY;
  else {
          TAtomo *pdel =  fila->frente;
          *x =  fila->frente->info;
           if ( filaUnitaria(fila) )
              InicializaFilaDinamica(fila);
           else
               fila->frente = fila->frente->prox;
           free(pdel);
      }
}
```

6.6 - Fila Dupla

A fila dupla ou deque é uma lista linear que tem como restrição de acesso, a inserção e a remoção dos elementos em ambas as extremidades. Elas podem ser organizadas em acesso estático (vectores) e acesso dinâmico (ponteiros). Nestas notas veremos apenas a forma de organização dinâmica.

6.7 - Fila Dupla Dinâmica

Declaramos uma fila dupla em organização dinâmica de forma análoga a declaração de uma fila simples. Devido as analogias entre essas estruturas, omitiremos a formalização das suas operações. Limitar-nos-emos a implementar as seguintes operações.

```
*/-------------------------------------------------------------------------------------------------------
Objectivo: Inserir um átomo com uma informação no fim do deque
Parâmetro Entrada: Deque  e uma determinada informação
Parâmetro de Saída:  Deque actualizado
Valor de Retorno: Correspondente código de erro.
------------------------------------------------------------------------------------------------------- */
 int  inserirDequeFim (TFilaDupla *fila, TInfo x)
{
   TAtomo  *pnovo = (Tatomo *)malloc(sizeof(TAtomo));
    if (pnovo == NULL)
         return NO_SPACE;
    else {
            pnovo->info =  x
            pnovo->dprox = NULL;
            pnovo->eprox = fila->fundo;
            if ( fila->fundo == NULL)
              {
                 fila->fundo = pnovo;
                 fila->frente = pnovo;
               }
            else
               {
                  fila->fundo->dprox = pnovo;
                  fila->fundo = pnovo;
               }
             return OK;
        }
}


*/-------------------------------------------------------------------------------------------------------
Objectivo: Remover o átomo que está no ínicio do deque
Parâmetro Entrada:  Um Deque
Parâmetro de Saída: Deque actualizado, informação removida
Valor de Retorno: Correspondente Código de código de erro.
------------------------------------------------------------------------------------------------------- */
int removerDequeInicio (TFilaDupla *fila, TInfo *x)
{
   if (vaziaDeque(fila))
      return FILA_EMPTY;
   else {
             TAtomo *pdel  = fila->inicio;
              *x = pdel->info;
               if ( filaUnitaria(fila) )
                  inicializaDeque(fila);
```

```
            else {
                    fila->frente = fila->frente->dprox;
                    fila->frente->eprox = NULL;
                  }
             free(pdel);
             return OK;
        }
}
```

6.8 - Exercícios

6.8.1- Desenvolva uma função que recebe como parâmetro de entrada uma fila estática e um determinado elemento. Inserir esse elemento no ínicio da fila.

6.8.2- Dado uma fila estática. Implemente as operações de inserção e remoção de um elemento de tal forma que essa fila passe a funcionar como uma pilha.

6.8.3- Desenvolva uma função que recebe como parâmetro de entrada duas filas estáticas e um determinado índice k. Inserir a primeira fila na segunda a partir da k_ésima posição.

6.8.4- Desenvolva uma função que recebe como parâmetro de entrada uma fila circular estática e uma posição k. Separar essa fila em duas de tal forma que o elemento que está na k_ésima posição fique na segunda fila.

6.8.5- Desenvolva uma função que recebe como parâmetro de entrada uma fila estática. Devolver essa fila na ordem inversa.

6.8.6- Desenvolva as operações de inserção e remoção de um elemento numa fila circular estática sem utilizar a varíavel que conta o número de elementos inseridos.

6.8.7- Uma fila dupla é uma variante do tipo abstracto de dados fila onde as inserções e remoções são feitas em ambas as extremidades. Desenvolva as operações standard que definem esse tipo abstracto de dados.

6.8.8- Uma fila dupla circular é uma variante do tipo abstracto de dados fila circular onde as remoções e inserções são feitas em ambas as extremidades. Desenvolva uma função que recebe como parâmetro de entrada uma fila dupla circular e uma determinada informação. Inserir essa informação no ínicio da fila.

6.8.9- Desenvolva uma função que recebe como parâmetro de entrada uma fila estática e devolve essa fila com os elementos ordenados.

6.8.10- Considere o estacionamento de comboios na forma de uma fila dupla, e n combiois numerados sequencialmente na entrada. É possível obter todas as permutações na saída? Se a resposta for negativa, dê exemplos de permutações, tão pequenas quanto possíveis, que não possam ser geradas.

6.8.11- Desenvolva um programa para ler um número indeterminado de valores inteiros. O valor zero finaliza a entrada de dados. Para cada valor lido, determinar se ele é um número par ou ímpar. Se o número for par incluí-lo na FILA PAR; caso contrário, incluí-lo na FILA ÍMPAR. Após o término da entrada de dados, retirar um elemento de cada fila alternadamente (iniciando-se pela FILA ÍMPAR) até que ambas as filas estejam vazias. Se o elemento retirado de uma das filas for um valor positivo, então incluí-lo em uma PILHA; caso contrário, remover um elemento da PILHA. Finalmente, escrever o conteúdo da pilha.

6.8.12- Implemente o sistema para a biblioteca usando o TAD fila. Cada livro deve ser representado por um registro que contêm os seguintes campos: Nome do livro, disponibilidade, fila de espera. Ao requisitar um livro, a pessoa entra na fila de espera se o livro não estiver disponível. Quando um livro fica disponível, o primeiro da fila de espera do livro deve receber o livro. Implemente as demais funcionalidades (cadastra livro, retira livro, etc.) que julgar necessária.

6.8.13- Existem partes de sistemas operativos que cuidam da execução dos programas. Por exemplo, em num sistema em tempo-compartilhado ("time-shared") existe a necessidade de manter um conjunto de processos numa fila, a espera de serem executados. Desenvolva um programa que seja capaz de ler uma série de pedidos para:

a. Incluir novos processos na fila de processo;
b. Retirar da fila o processo com o maior tempo de espera;
c. Imprimir o conteúdo da fila de processo em determinado momento.

Assuma que cada processo é representado por um registro composto pelo um número que identifica o processo e o tempo para esse processo ser processado.